Московский Авиационный Институт

(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

**Лабораторная работа №5 по курсу**

**«Операционные системы»**

**Тема работы**

**«Динамические библиотеки»**

Студент: Меркулов Фёдор Алексеевич

Группа: М8О-207Б-21

Вариант: 35

Преподаватель: Миронов Евгений Сергеевич

Оценка: \_\_\_\_\_\_\_\_\_\_\_

Дата: \_\_\_\_\_\_\_\_\_\_\_

Подпись: \_\_\_\_\_\_\_\_\_\_\_

Москва, 2022

**Содержание**

1. Репозиторий
2. Постановка задачи
3. Общие сведения о программе
4. Общий метод и алгоритм решения
5. Исходный код
6. Демонстрация работы программы
7. Выводы

**Репозиторий**

<https://github.com/WhatTheMUCK/OSi/tree/main/LR_5>

**Постановка задачи**

Требуется создать динамические библиотеки, которые реализуют определенный функционал. Далее использовать данные библиотеки 2-мя способами:

1. Во время компиляции (на этапе «линковки»/linking)
2. Во время исполнения программы. Библиотеки загружаются в память с помощью интерфейса ОС для работы с динамическими библиотеками

В конечном итоге, в лабораторной работе необходимо получить следующие части:

* Динамические библиотеки, реализующие контракты, которые заданы вариантом;
* Тестовая программа (*программа №1*), которая используют одну из библиотек, используя знания полученные на этапе компиляции;
* Тестовая программа (*программа №2*), которая загружает библиотеки, используя только их местоположение и контракты.

Провести анализ двух типов использования библиотек.

Пользовательский ввод для обоих программ должен быть организован следующим образом:

1. Если пользователь вводит команду «0», то программа переключает одну реализацию контрактов на другую (необходимо только для *программы №2*).
2. «1 arg1 arg2 … argN», где после «1» идут аргументы для первой функции, предусмотренной контрактами. После ввода команды происходит вызов первой функции, и на экране появляется результат её выполнения;
3. «2 arg1 arg2 … argM», где после «2» идут аргументы для второй функции, предусмотренной контрактами. После ввода команды происходит вызов второй функции, и на экране появляется результат её выполнения.

**Вариант 35**:

Контракт 1:

Подсчет площади плоской геометрической фигуры по двум сторонам

Float Square(float A, float B)

Реализация 1:

Фигура прямоугольник

Реализация 2:

Фигура прямоугольный треугольник

Контракт 2:

Отсортировать целочисленный массив

Int \* Sort(int \* array, int size)

Реализация 1:

Пузырьковая сортировка

Реализация 2:

Сортировка Хоара

Система сборки: CMake.

Вариант 5:

Типы чисел, используемых при вычислениях, в библиотеках меняются в зависимости от операционной системы, на которой происходит сборка.

Windows: без изменений

Linux: int -> long, float -> double

MacOS: int -> short, float -> long double

Сигнатуры функций остаются без изменений.

**Общие сведения о программе**

Программа состоит из двух интерфейсов (main1.cpp и main2.cpp), каждый из них реализован по-разному, в соответствии с заданием. Также каждая реализация контрактов представляет из себя отдельный файл: lib1.cpp и lib2.cpp. Для объявления необходимых функций также используется заголовочный файл lib.h, а для объявления типов: заголовочный файл types.h. Так как все собирается с помощью CMake, то в проекте присутствует CMakeLists.txt.

**Общий метод и алгоритм решения**

Объявим необходимые функции внутри файла lib.h. Используем спецификатор хранения extern, который сообщает компилятору, что находящиеся за ним типы и имена переменных объявляются где-то в другом месте.

Так как по заданию необходимо подключать библиотеки на этапе линковки, то подключать lib.h в реализации lib1.cpp и lib2.cpp не следует. В этих файлах просто напишем логику работы необходимых функций. Важно, чтобы они назывались также, как и те, что объявлены в lib.h.

Интерфейс 1:

Подключаем lib.h и пользуемся функциями так, как будто библиотека обычная. Различия наступают в сборке программы. Если бы мы собирали такой код в терминале, то прописали бы g++ -c -fpic lib1.cpp. Опция -fpic - требует от компилятора, при создании объектных файлов, порождать позиционно-независимый код. Формат позиционно-независимого кода позволяет подключать исполняемые модули к коду основной программы в момент её загрузки. Далее g++ -shared -o liblib1.so lib1.o -lm. Опция -shared - указывает g++, что в результате должен быть собран не исполняемый файл, а разделяемый объект — динамическая библиотека.

Интерфейс 2:

Воспользуемся системными вызовами из библиотеки <dlfcn.h>.

Функция dlopen открывает динамическую библиотеку (объект .so) по названию.

Функция dlsym - обработчик динамически загруженного объекта вызовом dlopen (мы передаём dlsym первым аргументов описатель динамического объекта, вызванного в dlopen).

Функция dlclose, соответственно, закрывает динамическую библиотеку.

Собираем с помощью g++ -L. -Wall -o main.out main2.cзз -llib2 -llib1. Флаг -L. Означает, что поиск файлов библиотек будет начинаться с текущей директории.

**Исходный код**

**types.h**

#include "stdlib.h"

#include "stdio.h"

#if OS == APPLE

typedef short os\_int;

typedef long double os\_float;

//#endif

#elif OS == UNIX

typedef long os\_int;

typedef double os\_float;

//#endif

#elif OS == WIN32

typedef int os\_int;

typedef float os\_float;

#endif

**lib.h**

#ifndef \_\_LIB\_H\_\_

#define \_\_LIB\_H\_

#include "stdlib.h"

#include "stdio.h"

#include "types.h"

/\*

#if OS == APPLE

typedef short os\_int;

typedef long double os\_float;

//#endif

#elif OS == UNIX

typedef long os\_int;

typedef double os\_float;

//#endif

#elif OS == WIN32

typedef int os\_int;

typedef float os\_float;

#endif

\*/

extern "C" os\_float Square(os\_float A, os\_float B);

extern "C" os\_int \*Sort(os\_int \*array, os\_int size);

#endif

**lib1.cpp**

#include <stdio.h>

#include <iostream>

#include "../include/types.h"

using namespace std;

extern "C" os\_float Square(os\_float A, os\_float B)

{

cout << "\nRectangle area: ";

return A \* B;

}

extern "C" os\_int \*Sort(os\_int \*array, os\_int size)

{

int i, j;

for (i = 0; i < size - 1; i++)

{

for (j = 0; j < size - 1 - i; j++)

{

if (array[j] > array[j + 1])

{

os\_int temp = array[j + 1];

array[j + 1] = array[j];

array[j] = temp;

}

}

}

cout << "\nSorting an array with a bubble: ";

return array;

}

**lib2.cpp**

#include <stdio.h>

#include <iostream>

#include "../include/types.h"

using namespace std;

extern "C" os\_float Square(os\_float A, os\_float B)

{

cout << "\nThe area of a right triangle: ";

return A \* B / 2;

}

void quickSort (os\_int \*array, os\_int low, os\_int high)

{

os\_int i = low;

os\_int j = high;

os\_int pivot = array[(i + j) / 2];

os\_int temp;

while (i <= j)

{

while (array[i] < pivot)

i++;

while (array[j] > pivot)

j--;

if (i <= j)

{

temp = array[i];

array[i] = array[j];

array[j] = temp;

i++;

j--;

}

}

if (j > low)

quickSort (array, low, j);

if (i < high)

quickSort (array, i, high);

}

extern "C" os\_int \*Sort(os\_int \*array, os\_int size)

{

quickSort(array, 0, size-1);

cout << "\nSorting an array using Hoare sorting: ";

return array;

}

**main1.cpp**

#include <stdio.h>

#include <iostream>

#include "../include/lib.h"

using namespace std;

int main(int argc, char const \*argv[])

{

cout << "Введите: [key] [arg1] ... [argN]\n";

cout << "Если вы хотите посчитать площадь фигуры: 1 [first side] [second size]\n";

cout << "Если вы хотите отсортировать массив: 2 [size] [array[0]] [array[1]] ... [array[size-1]]\n";

int key;

while(cin >> key){

if (key == 1){

os\_float A,B;

cin >> A >> B;

cout << Square(A,B) << "\n";

}

if (key == 2){

os\_int size;

cin >> size;

os\_int a[size];

os\_int \*b;

for (int i=0; i<size; i++)

cin >> a[i];

b = Sort(a, size);

for (int i=0; i<size; i++)

cout << b[i] << " ";

cout << "\n";

}

if (key != 1 and key != 2){

cout << "\nНеправильрный ключ\n";

}

cout << "\nВведите: [key] [arg1] ... [argN]\n";

cout << "Если вы хотите посчитать площадь фигуры: 1 [first side] [second size]\n";

cout << "Если вы хотите отсортировать массив: 2 [size] [array[0]] [array[1]] ... [array[size-1]]\n";

}

}

**main2.cpp**

#include <stdio.h>

#include <iostream>

#include <dlfcn.h>

#include "../include/types.h"

using namespace std;

const char\* lib1 = "./liblib1.so";

const char\* lib2 = "./liblib2.so";

int main(int argc, char const \*argv[])

{

cout << "Введите: [key] [arg1] ... [argN]\n";

cout << "Если вы хотите поменять метод вычислений, введите 0\n";

cout << "Если вы хотите посчитать площадь фигуры: 1 [first side] [second size]\n";

cout << "Если вы хотите отсортировать массив: 2 [size] [array[0]] [array[1]] ... [array[size-1]]\n";

int key;

int checker = 0;

void\* CurrentLib = dlopen(lib1, RTLD\_LAZY);

cout << "Библиотека: " << checker << "\n";

os\_float (\*Square)(os\_float A, os\_float B);

os\_int\* (\*Sort)(os\_int \*array, os\_int size);

Square =(os\_float (\*)(os\_float,os\_float)) dlsym(CurrentLib, "Square");

Sort =(os\_int\* (\*)(os\_int\*, os\_int)) dlsym(CurrentLib, "Sort");

while(cin >> key){

if (key == 0){

dlclose(CurrentLib);

if (checker == 0){

CurrentLib = dlopen(lib2, RTLD\_LAZY);

} else {

CurrentLib = dlopen(lib1, RTLD\_LAZY);

}

checker = !checker;

Square = (os\_float (\*)(os\_float, os\_float)) dlsym(CurrentLib, "Square");

Sort =(os\_int\* (\*)(os\_int\*, os\_int)) dlsym(CurrentLib, "Sort");

}

if (key == 1){

os\_float A,B;

cin >> A >> B;

cout << Square(A,B) << "\n";

}

if (key == 2){

os\_int size;

cin >> size;

os\_int a[size], \*b;

for (int i=0; i<size; i++)

cin >> a[i];

b = Sort(a, size);

for (int i=0; i<size; i++)

cout << b[i] << " ";

cout << "\n";

}

if (key != 0 and key != 1 and key != 2){

cout << "\nНеправильный ключ\n";

}

cout << "\nВведите: [key] [arg1] ... [argN]\n";

cout << "Если вы хотите поменять метод выислений, введите 0\n";

cout << "Если вы хотите вычислить площадь фигуры: 1 [first side] [second size]\n";

cout << "Если вы хотите отсортировать список: 2 [size] [array[0]] [array[1]] ... [array[size-1]]\n";

cout << "Библиотека: " << checker << "\n";

}

}

**CMakeLists.txt**

cmake\_minimum\_required(VERSION 3.22.0)

project(5 LANGUAGES CXX)

set(DYNAMIC\_SOURCE src/main2.cpp)

set(STATIC\_SOURCE src/main1.cpp)

set(LIBRARIES1\_SOURCE src/lib1.cpp)

set(LIBRARIES2\_SOURCE src/lib2.cpp)

set(LIBRARIES\_INCLUDE include/lib.h)

add\_library(

lib1 SHARED

${LIBRARIES\_INCLUDE}

${LIBRARIES1\_SOURCE}

)

add\_library(

lib2 SHARED

${LIBRARIES\_INCLUDE}

${LIBRARIES2\_SOURCE}

)

add\_library(

lib\_st1 STATIC

${LIBRARIES\_INCLUDE}

${LIBRARIES1\_SOURCE}

)

add\_library(

lib\_st2 STATIC

${LIBRARIES\_INCLUDE}

${LIBRARIES2\_SOURCE}

)

if (UNIX)

add\_definitions(-DOS=UNIX)

endif()

if (APPLE)

add\_definitions(-DOS=APPLE)

endif()

if (WIN32)

add\_definitions(-DOS=WIN32)

endif()

add\_executable(main2 ${DYNAMIC\_SOURCE})

add\_executable(main11 ${STATIC\_SOURCE})

add\_executable(main12 ${STATIC\_SOURCE})

target\_link\_libraries(main11 PRIVATE lib\_st1)

target\_link\_libraries(main12 PRIVATE lib\_st2)

target\_link\_libraries(main2 PRIVATE dl)

**Демонстрация работы программы**

papik@papik-VirtualBox:~/OSlaba51/build$ cmake ..

-- Configuring done

-- Generating done

-- Build files have been written to: /home/papik/OSlaba51/build

papik@papik-VirtualBox:~/OSlaba51/build$ make

[ 7%] Building CXX object CMakeFiles/lib1.dir/src/lib1.cpp.o

[ 14%] Linking CXX shared library liblib1.so

[ 14%] Built target lib1

[ 21%] Building CXX object CMakeFiles/lib2.dir/src/lib2.cpp.o

[ 28%] Linking CXX shared library liblib2.so

[ 28%] Built target lib2

[ 35%] Building CXX object CMakeFiles/lib\_st1.dir/src/lib1.cpp.o

[ 42%] Linking CXX static library liblib\_st1.a

[ 42%] Built target lib\_st1

[ 50%] Building CXX object CMakeFiles/lib\_st2.dir/src/lib2.cpp.o

[ 57%] Linking CXX static library liblib\_st2.a

[ 57%] Built target lib\_st2

[ 64%] Building CXX object CMakeFiles/main2.dir/src/main2.cpp.o

[ 71%] Linking CXX executable main2

[ 71%] Built target main2

[ 78%] Building CXX object CMakeFiles/main11.dir/src/main1.cpp.o

[ 85%] Linking CXX executable main11

[ 85%] Built target main11

[ 92%] Building CXX object CMakeFiles/main12.dir/src/main1.cpp.o

[100%] Linking CXX executable main12

[100%] Built target main12

papik@papik-VirtualBox:~/OSlaba51/build$ ./main11

Введите: [key] [arg1] ... [argN]

Если вы хотите посчитать площадь фигуры: 1 [first side] [second size]

Если вы хотите отсортировать массив: 2 [size] [array[0]] [array[1]] ... [array[size-1]]

1

2 3

Rectangle area: 6

Введите: [key] [arg1] ... [argN]

Если вы хотите посчитать площадь фигуры: 1 [first side] [second size]

Если вы хотите отсортировать массив: 2 [size] [array[0]] [array[1]] ... [array[size-1]]

1

0.25 0.01

Rectangle area: 0.0025

Введите: [key] [arg1] ... [argN]

Если вы хотите посчитать площадь фигуры: 1 [first side] [second size]

Если вы хотите отсортировать массив: 2 [size] [array[0]] [array[1]] ... [array[size-1]]

2

5

5 4 3 2 1

Sorting an array with a bubble: 1 2 3 4 5

Введите: [key] [arg1] ... [argN]

Если вы хотите посчитать площадь фигуры: 1 [first side] [second size]

Если вы хотите отсортировать массив: 2 [size] [array[0]] [array[1]] ... [array[size-1]]

papik@papik-VirtualBox:~/OSlaba51/build$ ./main12

Введите: [key] [arg1] ... [argN]

Если вы хотите посчитать площадь фигуры: 1 [first side] [second size]

Если вы хотите отсортировать массив: 2 [size] [array[0]] [array[1]] ... [array[size-1]]

1

2 3

The area of a right triangle: 3

Введите: [key] [arg1] ... [argN]

Если вы хотите посчитать площадь фигуры: 1 [first side] [second size]

Если вы хотите отсортировать массив: 2 [size] [array[0]] [array[1]] ... [array[size-1]]

1

0.25 0.01

The area of a right triangle: 0.00125

Введите: [key] [arg1] ... [argN]

Если вы хотите посчитать площадь фигуры: 1 [first side] [second size]

Если вы хотите отсортировать массив: 2 [size] [array[0]] [array[1]] ... [array[size-1]]

2

5

-5 4 3 2 1

Sorting an array using Hoare sorting: -5 1 2 3 4

Введите: [key] [arg1] ... [argN]

Если вы хотите посчитать площадь фигуры: 1 [first side] [second size]

Если вы хотите отсортировать массив: 2 [size] [array[0]] [array[1]] ... [array[size-1]]

papik@papik-VirtualBox:~/OSlaba51/build$ ./main2

Введите: [key] [arg1] ... [argN]

Если вы хотите поменять метод вычислений, введите 0

Если вы хотите посчитать площадь фигуры: 1 [first side] [second size]

Если вы хотите отсортировать массив: 2 [size] [array[0]] [array[1]] ... [array[size-1]]

Библиотека: 0

1

2 3

Rectangle area: 6

Введите: [key] [arg1] ... [argN]

Если вы хотите поменять метод выислений, введите 0

Если вы хотите вычислить площадь фигуры: 1 [first side] [second size]

Если вы хотите отсортировать список: 2 [size] [array[0]] [array[1]] ... [array[size-1]]

Библиотека: 0

2

4

3 -3 2 -3

Sorting an array with a bubble: -3 -3 2 3

Введите: [key] [arg1] ... [argN]

Если вы хотите поменять метод выислений, введите 0

Если вы хотите вычислить площадь фигуры: 1 [first side] [second size]

Если вы хотите отсортировать список: 2 [size] [array[0]] [array[1]] ... [array[size-1]]

Библиотека: 0

0

Введите: [key] [arg1] ... [argN]

Если вы хотите поменять метод выислений, введите 0

Если вы хотите вычислить площадь фигуры: 1 [first side] [second size]

Если вы хотите отсортировать список: 2 [size] [array[0]] [array[1]] ... [array[size-1]]

Библиотека: 1

1

2 3

The area of a right triangle: 3

Введите: [key] [arg1] ... [argN]

Если вы хотите поменять метод выислений, введите 0

Если вы хотите вычислить площадь фигуры: 1 [first side] [second size]

Если вы хотите отсортировать список: 2 [size] [array[0]] [array[1]] ... [array[size-1]]

Библиотека: 1

2

5

1 4 2 3 4

Sorting an array using Hoare sorting: 1 2 3 4 4

Введите: [key] [arg1] ... [argN]

Если вы хотите поменять метод выислений, введите 0

Если вы хотите вычислить площадь фигуры: 1 [first side] [second size]

Если вы хотите отсортировать список: 2 [size] [array[0]] [array[1]] ... [array[size-1]]

Библиотека: 1

**Выводы**

В ходе лабораторной работы я познакомился с созданием динамических библиотек в ОС Linux, а также с непосредственной работе с ними, то есть познакомился с возможностью загружать эти библиотеки в ходе выполнения программы.